Impact of Adverse Network Conditions on P2P-TV Systems: Experimental Evidence

Eugenio Alessandria, Massimo Gallo, Emilio Leonardi, Marco Mellia[*], Michela Meo

Politecnico di Torino
Dipartimento di Elettronica
Corso Duca degli Abruzzi 24, 10129 - Italy

Abstract

In this paper we define an experimental set-up to analyze the behavior of commercial P2P-TV applications under adverse network conditions. Our goal is to reveal the ability of different P2P-TV applications to adapt to evolving network conditions, such as delay, loss, available capacity, and presence of background traffic and to check whether such systems implement some form of congestion control. We apply our methodology to four popular commercial P2P-TV applications: PPLive, SOPCast, TVants and TVUPlayer. Our results show that all the considered applications are in general capable to cope with packet losses and to react to congestion arising in the network core. Indeed, all applications keep trying to download data by avoiding bad paths and carefully selecting good peers. However, when the bottleneck affects all peers, e.g., it is at the access link, their behavior results rather aggressive, and potentially harmful for both other applications and the network.

We then observe the interference between TCP and P2P-TV traffic. As expected, P2P-TV applications do not perform TCP-friendly congestion control, causing in some cases problems to TCP traffic and to P2P-TV performance itself. Finally, we also verify that the applications are fair towards clients sharing

[*]Corresponding Author: Marco Mellia, Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10124 - Torino, Italy email: mellia@tlc.polito.it, tel: +39-011-090-4173, Fax: +39-011-090-4099

Email addresses: `leonardi@tlc.polito.it` (Emilio Leonardi), `mellia@tlc.polito.it` (Marco Mellia), `michela@tlc.polito.it` (Michela Meo)

the same access link, even when congestion arises.

Keywords:
Peer-to-Peer, Video Streaming Measurement

1. Introduction and Motivations

A new class of peer-to-peer (P2P) systems providing real-time video streaming over the Internet is fast emerging and gaining popularity. Several commercial P2P streaming systems such as PPLive, SOPCast, TVants and TVUPlayer, just to mention the most popular ones, are already available on Internet.

P2P-TV systems may contribute to revolution the broadcast TV paradigm allowing ubiquitous access to a practically unlimited number of channels. This represents an important step forward in the direction of an Anything-Anyone-Anywhere-Anytime ubiquitous communication paradigm of future Internet applications [1].

The adoption of a P2P paradigm reduces the network costs, pushing complexity from the network to the users, while helping to relieve the bandwidth cost burden at the servers. Although from the users' as well as from the server points of view this class of P2P applications has useful and interesting characteristics, the network operators' have serious concerns about the capability of the Internet to support large scale P2P-TV systems (mainly due to the potential high bandwidth requirements, large number of involved users, and the intrinsic inelasticity of video traffic). These concerns are confirmed by some news reports, see for instance [2]. It seems, therefore, rather urgent to have a better understanding of the potential impacts that these applications may entail on the underlying transport network. Since the most widely deployed commercial systems follow a closed and proprietary design, only an experimental and black box characterization of traffic injected by such systems is in general possible; we emphasize, indeed, that approaches requiring to partially reverse engineering complex P2P-TV systems are viable at much larger cost, and only in a few cases. Moreover, to develop new architectures and algorithms that improve the "network friendliness" of such applications [3, 4], it is necessary to understand how current applications react to different network conditions and scenarios. Do they implement any congestion control algorithm? How do they react to packet drop? What is the impact of increased end-to-end delay?

In this paper we propose a testing methodology and test-bed experiments to assess how these applications react to different network conditions, like available bandwidth, loss probability, delay; both network load, and user perceived quality of service, should then be measured. Applying our methodology, we test and compare four popular P2P-TV applications, namely PPLive, SOPCast, TVants and TVUPlayer. All selected applications adopt the “mesh-based” approach [1], in which peers form a generic overlay topology to exchange chunks of data.

Results show that all applications are effective in trying to overcome network impairment. For example, all applications avoid impaired paths by carefully selecting peers to download from. However, when the bottleneck affects all paths, e.g., in case the access link is congested, they aggressively download data trying to receive the video stream. While P2P-TV offers good end-user service even in presence of adverse network conditions and it is far towards clients sharing the same access link, it can become harmful to the network and other applications. In particular, the coexistence of these applications with elastic background traffic, e.g., TCP connections, can result critical in some cases. We discover then that all applications implement a memory based algorithm that tracks good and bad neighbor peers, while no change is observed in the mechanisms to create the neighbors set and that applications.

The paper is organized as follows. We start by summarizing the related work in Sec. 2. Then, the measurement setup and methodology are defined in Sec. 3. Sec. 4 presents the results in which network impairment affects the incoming traffic from all peers, in Sec. 5 scenarios in which “good” and “bad” peers coexist are analyzed to investigate the ability of the applications to correctly handle them; scenarios in which the P2P-TV applications are sharing the access bandwidth with TCP connections are instead analyzed in Sec. 6 and the case of two clients sharing the same access link is made in Sec. 7. Finally, Sec. 8 summarizes our findings.

2. Related Work

To the best of our knowledge, this is the first experimental work on P2P-TV systems exploring how such systems react to different network conditions. In a previous paper, we performed a similar characterization considering Skype [5], in which the focus was on the voice traffic sent/received by a Skype client.

Considering more general experimental results about P2P-TV systems, the research community has given a lot of attention to understand application internals [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].

A few works [6, 7, 8], relying on the implementation of an active crawler, focus on a single system. These approaches face the daunting task of partially reversing the engine of the P2P-TV system under analysis. As a consequence, this methodology is limited by the ability to break closed and proprietary systems, and we believe that they can be hardly extended to characterize all the possible P2P-TV applications. In particular, [6] investigates PPLive, whereas [7] focuses on the commercial re-engineer of Coolstreaming, and [8] considers UUSee. All these papers mainly provide a big picture of the considered system, focusing on metrics such as the number of users, their geographical distribution, the session duration, the packet size distribution. None of the above mentioned papers considers the particular aspects we are interested into, i.e., the way the system reacts to evolving network conditions.

Other works, such as [9, 10], instead, study specific aspects of a P2P streaming system. For instance, [9] gives some preliminary results on the node degrees of popular versus unpopular channels in PPLive. Authors in [10] investigate the stability of nodes in PPLive, and devise schemes to identify the most stable nodes.

Quality of service is of concern in [11, 12]. Authors in [11] exploit an analysis of PPLive buffer maps, collected through protocol reverse engineering, to infer QoS metrics such as network-wide playback continuity, startup latency, playback lags among peers, and chunk propagation timing. Authors in [12] focus on similar metrics but exploit logs made available from an (unspecified) commercial P2P streaming system.

Authors in [13] analyze and compare PPLive and SOPCast investigating the time evolution of different metrics, like transmitted/received bytes, number of parents and children, etc. In [14], on the contrary, a comparative evaluation of four commercial systems (namely PPLive, PPStream, SOPCast and TVAnts) is proposed showing flow-level scatter plots of mean packet size versus flow duration and data rate of the top-10 contributors versus the overall download rate. In [15] PPLive, SOPCast and TVAnts systems are analyzed. A systematic exploration of the mechanisms driving the peer-selection in the different systems is performed. At last, in [16] a simple experimental analysis of PPLive and Joost is presented to evaluate the characteristics of both data distribution and signaling process for the overlay network discovery and maintenance.

A preliminary version of this paper has been presented at Infocom 2009, [17]. In this paper, we extend the analysis of [17] by investigating the transmitted traffic, the coexistence of P2P-TV and TCP traffic, and fairness towards clients sharing the same access link. Some results of the previous version are omitted here, to leave room for the new results.

3. Methodology

The aim of this work is to study how P2P-TV applications react to different network scenarios. Given that all successful P2P-TV applications follow a proprietary and closed design, we have to follow a “black-box” approach. We therefore setup a testbed, in which clients running the Application Under Test (AUT) are connected to a Linux router, which is connected to the Internet via our Fast-Ethernet based campus LAN. The router itself is then used to enforce particular network conditions. In particular, we used the Linux Network Emulation functionality `netem` coupled with the Token Bucket Filter `TBF`. This allows us to emulate the properties of wide area networks, controlling available bandwidth, delay, loss, duplication and re-ordering of packets routed through the router. Other PCs are connected to the testbed to inject background traffic.

Note that, since we run real on-field experiments, our control on the experimental set-up is limited to the interface in object only. This implies that the global network conditions are unknown and that possible effects due to congestion, loss, delay inside the Internet are superposed to the effects “artificially” introduced at the router under our control.

Two packet level traces are collected at the router: the first one logs all packets sent/received by the network interface that connects the router to the Internet; the second one logs all packets sent/received by the network interface that connects the PC running the AUT. Packet level traces are then post-processed to obtain the desired measurements. In this paper, we report results considering the *average received bit-rate* measured in small time windows (set to 1 minute); the received bit-rate is denoted by $r(t)$, with t the time at the end of the measurement window, and it is evaluated at the application layer, i.e., neglecting transport, network and data-link overheads. Similarly, in some cases we report the *average transmitted bit-rate* $s(t)$, intended as the bit-rate transmitted by the peer interface; $s(t)$ is also measured in 1 minute long intervals. The number $n(t)$ of peers that exchanged packets with the AUT during a time interval, i.e., the *number of active peers*, is evaluated as well; for $n(t)$ we use shorter time windows of 5 seconds.

Finally, the PC running the AUT is used to capture the video stream that is received and to dump it on a file by means of a video grabber utility. To evaluate the video quality of the received stream, we cannot apply any standard reference-based technique, since they all rely on the comparison of the received and original video (being impossible to get the latter one). All selected applications generate 378 kbps streams encoded using the Microsoft VC-1 encoder; a typical bit-rate of 450-500 kbps is received by the AUT, so that 100-150 kbps of additional overhead is required by the applications to successfully deliver the stream (not including transport, network and data-link headers). Since the codec relies on proprietary design, it is difficult to evaluate the quality of the received stream. We are therefore forced to estimate the stream quality by simply counting the number of errors a decoder has to deal with when decoding the stream. In particular, we decoded each file using `ffmpeg` utility which reports a detailed list of corrupted video I-frames. Those are major impairment that will affect the video quality for several frames, i.e., up to when a good I-frame is received (usually several seconds later). Similarly, the audio stream decoding errors are evaluated as reported by `ffmpeg`. In the following, we report therefore the number of corrupted I-frames and audio blocks as quality index. While this allows only a qualitative evaluation of the stream quality, it allows us to fairly compare different applications.

3.1. Scenario definition

The parameters we consider in this paper are the following:

- c: Capacity limit
- l: Packet loss
- d: Delay

The set $L(t) = \{c(t), l(t), d(t)\}$ specifies the state of the controlled link during each instant of the experiment - we restrict to the cases in which only one of the three above parameters is evolving with time and we denote with $p(\cdot)$ its *profile* over time.

As profile $p(\cdot)$ we select a step function, with initial value p_0, increments I, and step duration ΔT, so that

$$p(t) = p_0 + I \sum_{n=1} H(t - n\Delta T) \tag{1}$$

in which $H(t)$ is the Heaviside step function

$$H(t) = \begin{cases} 0 & t < 0 \\ 1 & t \geq 0 \end{cases} \tag{2}$$

If I is positive, $p(\cdot)$ corresponds to an *increasing* profile; negative values of I, on the contrary, generate *decreasing* profiles. A null increment, $I = 0$, finally, leads to a *constant* profile.

The impairment defined by a scenario can affect all sent/received packets, so that *global* impairment is imposed, or only a subset of sent/received packets, so that *per peer* impairment is imposed; for example the scenario can affect a single peer, a subnet, an Autonomous System, or any generic subset of IP addresses.

3.2. Considered general setup

We performed several experiments considering different scenarios and profiles, during various time periods and with clients tuned on different TV channels. We collected a total of more than 300 hours of experiments. In this paper, we report a subset of the most representative experiments. In particular, we consider only scenarios in which the download link is controlled, while the upload link capacity is limited to $c_u(t) = 200$ kbps, unless otherwise specified. This indeed allows us to evaluate the application behavior when the peer has not enough capacity to act as an "amplifier", i.e., to serve many peers; this is the typical condition of ADSL users.

4. Global impairment

4.1. Effect of available capacity

In the first set of experiments, the download available capacity $c(t)$ is imposed. Results are shown in Fig. 1 and are organized in the following way. The two largest plots in the left part of the figure report the bit-rate $r(t)$ versus time for profiles with either decreasing or increasing capacity limit (on the top and bottom plots, respectively). The 8 small plots on the right part of the same figure depict the number of corrupted audio and video frames for the same experiments; each plot refers to a specific application.

Let us start by considering the decreasing profile. Every $\Delta T = 5$ minutes, the available bandwidth is decreased by a $I = -50$ kbps, starting from an initial value of $c_0 = 800$ kbps. The average bit-rate evaluated using 60 seconds time intervals is reported for all applications on left top plot of the figure. The experiment lasted 1 hour, after which the available capacity was set back to 800 kbps. All applications have similar behavior: the bit-rate remains basically constant for all the time the available capacity is larger than the data rate, $c(t) > r(t)$. When the capacity bottleneck kicks in, all the applications react by increasing the download data rate. Consider, for example, TVAnts, which exhibits the largest value of the bit-rate. The normal data rate is about 600 kbps; when the capacity limit reaches 650 kbps, the receiver starts suffering the bottleneck (due to traffic burstiness), and it reacts by requesting other peers to send more traffic; the download rate becomes

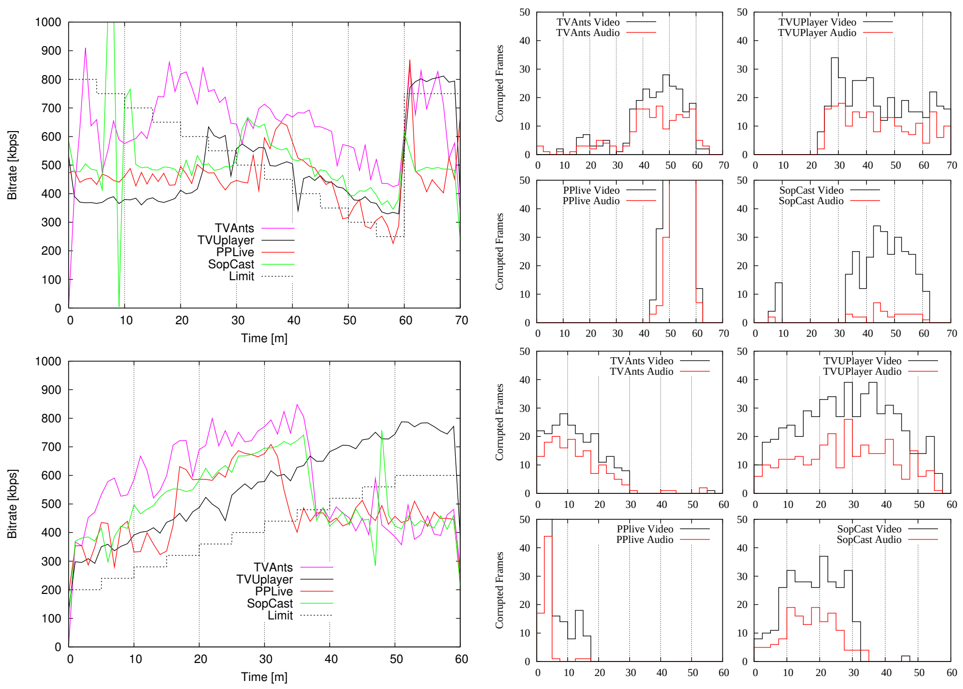

Figure 1: Left plots: received bit-rate for decreasing (top) and increasing (bottom) available bandwidth. Right plots: percentage of corrupted audio/video frames, one plot for each application.

larger than 800 kbps. As the capacity limits $c(t)$ decreases, the received rate decreases too, being always about twice the available capacity, i.e., the offered load to the congested link is about 2, $r(t)/c(t) \simeq 2$. Other applications show similar behavior, with smaller values of the offered load in congested conditions; in particular, TVUPlayer exhibits the smallest overload factor $r(t)/c(t) \simeq 1.3$.

Looking at the last 10 minutes of the experiment, when the capacity returns to high values, an unexpected, strange behavior is observed. Indeed, since $c(t)$ is larger then the normally required capacity, $r(t)$ should take again the typical values that can be observed when traffic is not bottlenecked. While this is true for PPLive and SopCast, both TVAnts and TVUPlayer keep on receiving at a rate which is about twice as large as the normal receiver rate. This maintains the bottleneck offered load higher than 1, so that audio/video impairment is observed up to the end of the experiment. Indeed, looking at the number of corrupted frames reported in the top right part of Fig. 1, audio/video impairment starts to show up as soon as the bottleneck kicks in, and it does not always disappear when the bottleneck capacity is set back to 800 kbps. TVAnts shows the longest period during which corrupted frames are observed, while, PPLive can cope with downlink capacity as small as 400 kbps without any audio/video error.

Results for the case of an increasing capacity profile are reported in bottom part of Fig. 1. The AUT is started now in scarce bandwidth conditions ($b_0 = 200$ kbps), and $I = 40$ kbps increments are applied every $\Delta T = 5$ minutes. All applications react to the adverse condition by trying to download much more data than the available capacity; also in this case $r(t)/c(t)$ varies from 1.3 (for TVUPlayer) to 2 (for TVAnts). Only when the available capacity is large enough to sustain the minimal download rate, all applications but TVUPlayer decrease $r(t)$ to their typical values. This is reflected by the disappearance of audio/video impairment, as shown by the right plots. Again, TVUPlayer suffers major impairment, even when the bandwidth grows to large values.

We can conclude that P2P-TV applications do not correctly perform congestion control, in scenarios in which peer access links get congested. They all try and react to limited access capacity by increasing the redundancy (by FEC or ARQ mechanisms) and, thus, the download rate. This may be potentially harmful for both the network and other applications sharing the congested link. Note that a single congested link may also be present when the unique peering link between a stub ISP and the rest of the Internet is congested. If this happens, P2P-TV applications may react as in the previously presented cases, causing further network problems and congestion.

4.2. *Effect of loss probability*

The second set of results we report aims at investigating the impact of loss probability on the AUT. Organized in a similar way as the previous figure, Fig. 2 shows the receiver rate for increasing (top plots) and decreasing (bottom plot) packet loss probability profiles. The right y-axis of the larger plots reports the percentage of losses, $l(t)$, that varies in time steps of $\Delta T = 5$ minutes, with loss increment $I = 5\%$ ($l_0 = 0\%$). In this case also, all the applications react to increasing packet losses by increasing the bit-rate $r(t)$. By doubling its received data rate for $l(t) > 35\%$, TVUPlayer is the most aggressive application, while PPLive shows the smallest increase. Looking at the corresponding number of audio/video corrupted frames, it is impressive to observe that all applications achieve very good video quality as far as $l(t) < 25\%$. In particular, it is worth noticing that SopCast can cope with 25% packet loss probability with only about 100 kbps of additional data rate. TVUPlayer exhibits similar performance, but at a much higher cost that accounts to up to 600kbps of additional data rate.

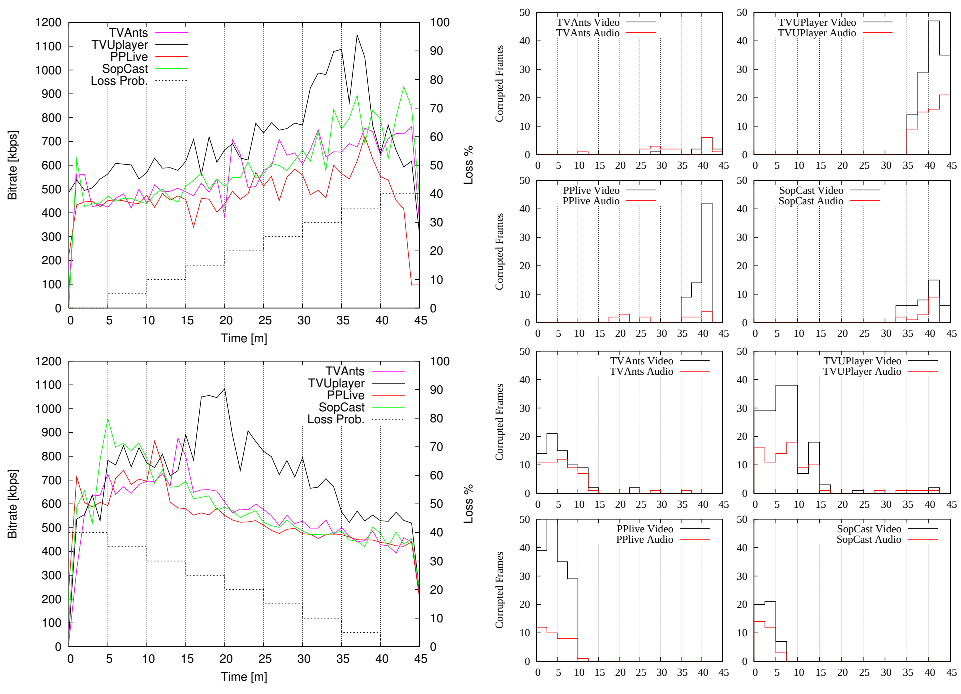

Figure 2: Left plots: received bit-rate for increasing (top) and decreasing (bottom) loss probability. Right plots: percentage of corrupted audio/video frames, one plot for each application.

Similar observations can be drawn by looking at the decreasing packet loss probability scenario reported in bottom plots of Fig. 2. In this scenario, $I = -5\%$, $i_0 = 40\%$, $\Delta T = 5$ minutes.

These results allow us to conclude that all applications react to packet losses by trying to recover them, using some kind of ARQ mechanism that causes an increase of the received traffic. While this is very efficient in repairing the audio/video stream, it comes at the expense of an offered load that can be as large as twice the rate in normal conditions. This definitively confirms that P2P-TV applications do not perform, in general, any congestion control.

4.3. *Effect of delay*

We now consider the effect of increasing and decreasing delay profiles. Fig. 3 reports the results for the received bit-rate of the increasing (left plot, $I = 200$ ms, $d_0 = 0$ ms, $\Delta T = 5$ minutes) and decreasing (right plot, $I = -200$ ms, $d_0 = 2000$ ms, $\Delta T = 5$ minutes) profiles; plots about the number of corrupted frames are not reported for the sake of brevity.

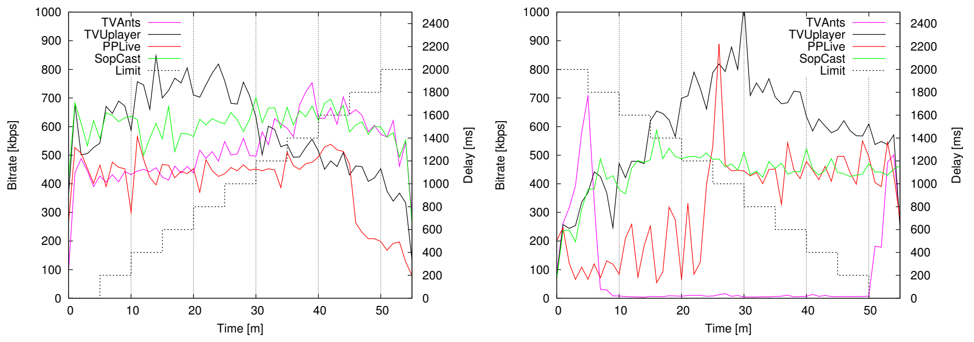

Figure 3: Received bit-rate for increasing (left plot) and decreasing (right plot) delay.

Results about the increasing delay case show that the applications can manage quite well slow variations of the delay; they can stand up to 1.5 s of additional delay without any significant variation of the received bit-rate (and any audio/video error). The applications start suffering the delay when it reaches almost 2 s, which is quite large; PPLive and TVUPlayer seem the most delay sensitive applications.

Interestingly, the applications suffer more for large values of the delay at the start up (see right plot Fig. 3). It is probably difficult for the applications to successfully create the neighbor list. Indeed, since the additional delay applies also to packets carrying signaling information, signaling dialogs are probably hardly completed with large values of the delay. In the decreasing profile, delay has to decrease below 1.2 s to allow the applications to start receiving the video stream. Again, PPLive seems the most sensitive application: additional delay should be smaller than 1 s to allow it to work.

4.4. Number of active peers

Finally, upper plots of Fig. 4 report the number of active peers, $n(t)$, for the previously described scenarios with increasing and decreasing capacity limit and loss probability.

The network conditions have no impact on the behavior of $n(t)$, which repeats regularly during the whole experiment. On the contrary, different experiments show different absolute values of $n(t)$; indeed, the absolute values change with channel popularity and time of the day so as to reflect the population of available peers. PPLive, that is an extremely popular application, has always the largest number of active peers.

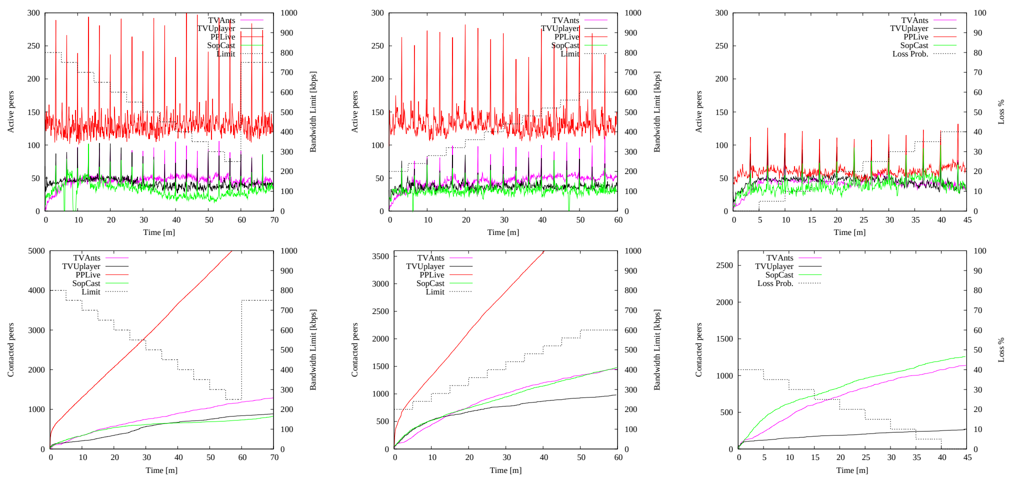

Figure 4: Number of active peers (top) and total contacted peers (bottom) for decreasing or increasing available bandwidth and packet loss probability (from left to right).

The same observations can be made by considering the evolution of the number of total contacted peers, which is independent from the network conditions; the associated plots are reported in the bottom part of Fig. 4. Notice also that the periodic peaks clearly visible in most of the applications are due to periodic keep-alive messages used to exchange signaling information.

These results allow us to conclude that the internal algorithms each application implements to discover, create and maintain the overlay, are insensitive and do not adapt to network conditions; network conditions influence only the video stream distribution mechanisms.

5. Per peer impairment

5.1. Effect of available capacity

We now investigate the capability of the AUT to cope with scenarios in which only a subset of peers is affected by network impairment, so that "good" and "bad" peers coexist. The goal is to verify if the AUT can identify the set of "good" peers to download from. In particular, for the results reported here, the imposed network impairment affects only peers having an odd IP address. The rationale behind this choice is to have the peer population split into two equally large subsets: odd peers, affected by network impairment, and even peers, not affected.

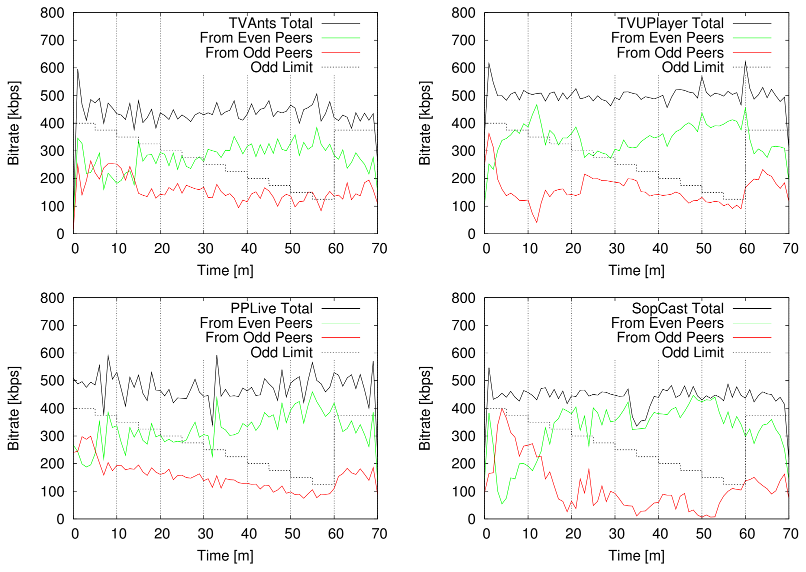

Figure 5: Received bit-rate for decreasing capacity limit. Odd peers only are affected by the impairment.

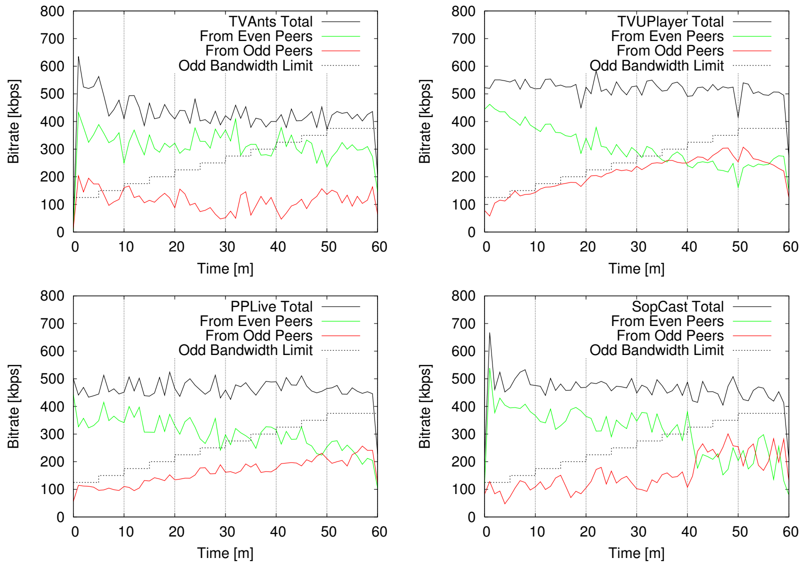

Figure 6: Received bit-rate for increasing capacity limit. Odd peers only are affected by the impairment.

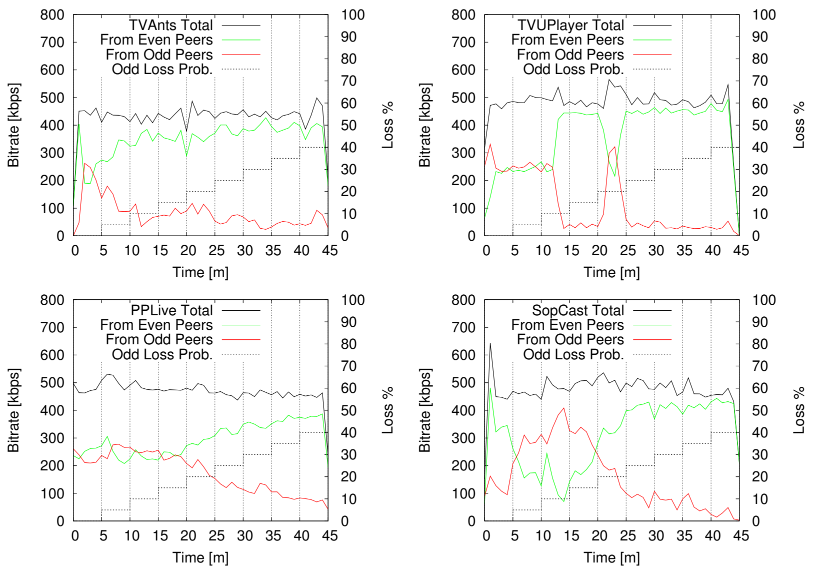

Figure 7: Received bit-rate for increasing loss probability. Odd peers only are affected by the impairment.

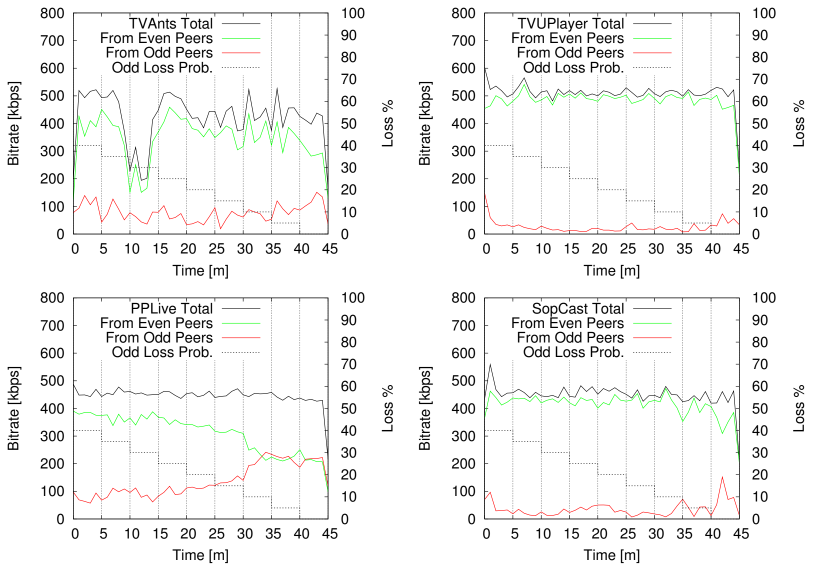

Figure 8: Received bit-rate for decreasing loss probability. Odd peers only are affected by the impairment.

The plots of Fig. 5 report results considering a decreasing capacity limit profile. In particular, the profile $c(t)$, that is imposed to odd peers only, starts from $c_0 = 400$ kbps, and every $\Delta T = 5$ minutes a further bandwidth decrease of $I = -25$ kbps is applied. After 60 minutes, the available bandwidth is again set to the initial value. Each plot reports, for a given application, the bit-rate received from even and odd peers and the total received bit-rate. The imposed profile is also given for completeness. Again, all applications exhibit similar behavior: during the initial phase there is no preference in receiving data from even or odd peers: they equally contribute to the total download rate. As soon as the bandwidth limit kicks in, reducing the performance of odd peers, the applications preferentially download data from even peers. The preference is stronger for SopCast (rightmost plots) for which even peers contribute to 80-90% of download rate. TVAnts, on the contrary, adapts less than the other applications to these network conditions.

Fig. 6 reports results for an increasing bandwidth profile applied to odd peers only; the profile has parameters: $c_0 = 125$ kbps, $I = 25$ kbps, $\Delta T = 5$ minutes. Similar considerations hold: All applications quickly identify the adverse capacity constraints affecting odd peers, so that even peers provide larger contribution to the total download bit-rate. In particular, TVUPlayer (top right plot) has a very accurate control mechanism that allows it to quickly identify the changing network conditions. PPLive and SopCast also exploit the additional bandwidth of odd peers that becomes gradually available, but a longer transient phase is required. Finally, TVAnts ignores the additional bandwidth, since about 70% of traffic is received from even nodes during the whole experiment.

In all cases, all the applications receive the minimum required amount of data that guarantees them to decode the audio/video streams without suffering any error.

5.2. Effect of loss probability

Figs. 7 and 8 report results considering increasing and decreasing profiles of $l(t)$, respectively. Let us start by considering increasing loss probability; plots on top row of the figure refer to a profile with $l_0 = 0$%, $I = 5$%, $\Delta T = 5$ minutes. In this case, different reactions are observed. TVAnts has a strong preference to receive from even peers starting from $l(t) \geq 5$%. TVUPlayer has an on/off behavior, so that no preference is shown up to $l(t) = 10$%, and, then, 95% of data is received from even nodes only. PPLive is ignoring the loss impairment up to $l(t) \geq 20$%, after which data are preferentially received from even nodes (but still 20% of traffic is received from odd peers when $l(t) = 40$%). Finally, SopCast shows a more irregular and uncontrolled behavior which causes a preference toward odd peers until $l(t) = 15$%, after which about 90% of data is received from even nodes only.

Consider now Fig. 8, which reports results for a decreasing profile of $l(t)$ (with $l_0 = 40$%, $I = -5$%, $\Delta T = 5$ minutes). Since all applications start in very unfavorable conditions for odd nodes, most of the traffic is received from even nodes. In particular, TVUPlayer constantly receives 98% of traffic from even nodes only, even when $l(t)$ becomes small. Similarly, TVAnts and SopCast exhibit a very stable preference during the whole experiment duration, with TVAnts trying to received 15-20% of traffic from impaired peers. PPLive, on the contrary, keeps on receiving 20% of traffic from odd nodes, percentage that goes up to 50% when $l(t) \leq 10$%. This confirms that PPLive is capable of coping with high packet loss rates (as already noticed in Fig. 2), hinting to an effective FEC algorithm.

5.3. Other results

We performed other similar tests, targeting with impairment: a particular peer, IP subnetworks, and Autonomous Systems. All the experiments showed consistent results, so that preference is given to good peers. We, thus, conclude that all applications implement a per-peer preference mechanism that is used to select the subset of good performing peers. While internal algorithms are unknown, the presented results suggest that the applications are using different algorithms. Due to space constraint, we do not report the figures referring to other scenarios we tested, and refer the reader to [17, 18] for more details.

Considering other possible impairment, experimental results shows that:

- **Delay preference:** All applications are very sensitive to additional delay, so that content was almost exclusively retrieved from good peers as soon as the additional delay increased.

- **Hop distance:** When we artificially decreased the IP TTL value to artificially inflate the path hop distance, no application showed appreciable bias. This clearly indicates that hop-count information is not exploited by the peer selection algorithm.

- **Number of active peers:** Results show that the AUT keeps contacting odd peers, since $n(t)$ is not correlated with $c(t)$, $l(t)$ or $d(t)$. This hints to control algorithms that react to different network scenarios by carefully selecting the good peers to exchange data with. However, signaling is exchanged with all peers (including “bad” peers) independently from the instantaneous

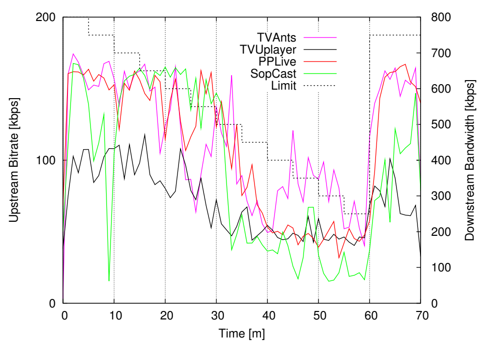

Figure 9: Transmitted bit-rate for decreasing bandwidth limitation.

end-to-end network quality. We also verified that all AUTs keep exchanging data with and probing "bad" peers even during very unfavorable conditions. In these cases only few, small packets are sent and (possibly) received. This suggests that only signaling information is exchanged between any two peers that are experiencing adverse network conditions, but the bad peers are not dropped in favor of the good ones.

5.4. Impact on the upstream traffic

In this section, we consider the impact of downlink impairment on uplink traffic. As before, we consider an ADSL-like scenario in which the peer has upstream capacity equal to $c_u = 200$kbps, i.e., the peer contribution to video distribution is low.

Figure 9 shows the total bitrate in the upstream for the considered applications when the downlink undergoes the same bandwidth decreasing profile previously presented. When the impairment on the downlink becomes severe, the peer reduces the number of chunk transmissions, that is, it reduces its contribution to the video content distribution. The transmitted information on the uplink is then basically limited to the signaling information.

To further investigate this phenomenon, we separately consider signaling and data information. Similar to what is done in [6, 16], signaling and data information is distinguished based on the packet size: packets whose size is smaller than 400 B are assumed to carry signaling information, larger packets carry data. Fig. 10 reports the size of the observed packets for the four applications; red and

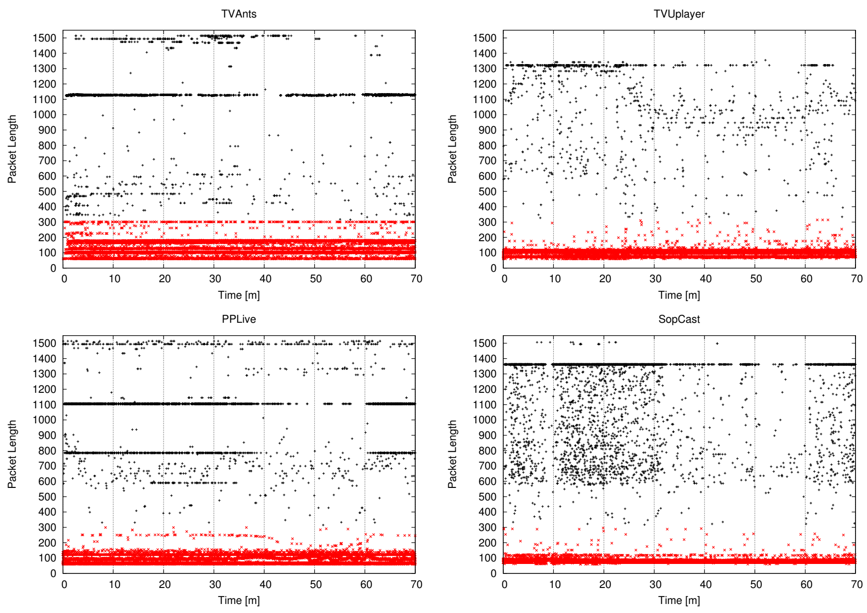

Figure 10: Packet size for decreasing bandwidth: signaling and data information.

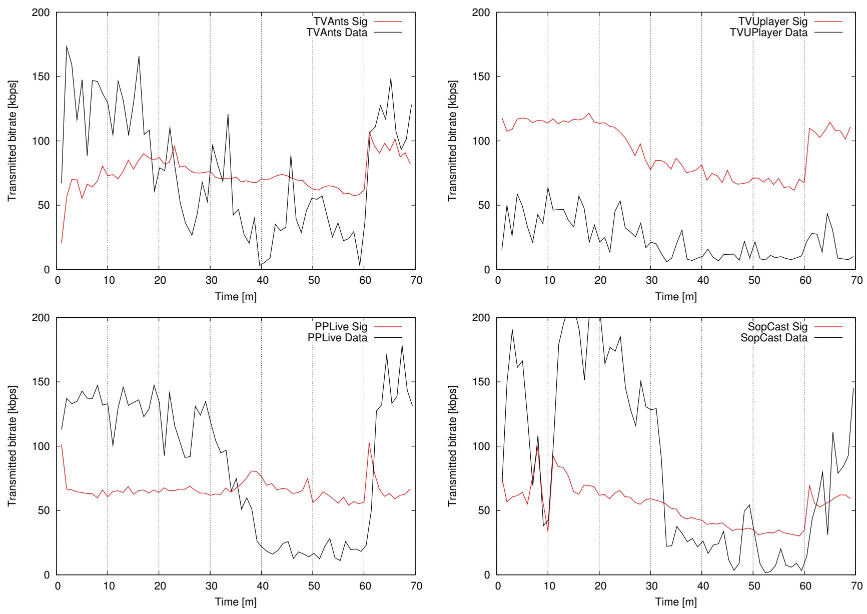

Figure 11: Transmitted bit-rate for decreasing bandwidth: signaling and data information.

black points refer to signaling and data packets, respectively. Information packets are transmitted in the first part of the experiment, when the downlink bandwidth limitation is not severe, so that the peer can retransmit video chunks to its neighbors. When downlink limitation becomes so strong that the peer is perceived by its neighbors as poor performing, the peer reduces its contribution to video distribution (in the figure, few data packets are transmitted when severe downlink limitation applies). On the contrary, the bandwidth limitation profile marginally affects signaling information, whose pattern is basically unchanged for the whole duration of the experiment. Indeed, the signaling information that is needed by the peer to guarantee the correct reception of the video is continuously transmitted, regardless the peer performance. The same observations can be derived by focusing on the signaling and data information bit-rate, reported in Fig. 11. For all the applications, signaling information requires more or less the same amount of bandwidth during the whole experiment, while data information diffusion depends on the network conditions (i.e., downlink bandwidth limitation). It is interesting that all the applications require roughly the same amount of signaling information, between 50 and 100 kbps, Sopcast being the application with the lowest signaling overhead.

6. Sharing the access bandwidth with TCP flows

In this section we analyze the effects of possible access bandwidth competition between persistent downstream TCP flows and P2P-TV applications. We consider a static setup in which the downstream access capacity is constant and equal to 2 Mbit/s, while the upstream bandwidth is, as usual, limited to 200 kbit/s. The P2P-TV application is running from $t = 0$, while a total of 5 TCP flows are started, one every $\Delta T = 300$ s. In particular, a PC connected on the same subnet of the P2P-TV client starts downloading a 5 GB long file from a Web server connected immediately after the Linux router.

Fig. 12 shows the results for the four considered applications, reporting both the average download and upload bitrate for the P2P-TV application, and the total TCP throughput. First observe that the received bit rate $r(t)$ is mainly insensitive to the presence of concurrent traffic for all the four applications. This behavior is expected in light of the fact that P2P-TV applications are non-elastic. More surprisingly, TCP connections fail to efficiently exploit the available bandwidth. In the case of TVAnts and PPlive, in particular, the TCP connections are severely impacted by the presence of P2P-TV traffic, so that the available downlink bandwidth cannot be successfully exploited.

This phenomenon is explained by observing the upstream channel, where congestion arises. Indeed, the P2P-TV application keeps uploading traffic, so that the narrow 200 kbit/s channel becomes congested, impairing TCP ACKs sent by receivers. Congestion on the upstream channel therefore induces ACK losses, preventing TCP from effectively exploiting available capacity on the downlink path. Notice indeed that the average loss probability in the link is about 10%.

To better understand the interaction between TCP flows and P2P-TV systems, in Fig. 13 we also consider a second scenario in which the upload bandwidth has been increased to 10 Mbit/s. The figure shows that TCP flows succeed in efficiently exploiting the available bandwidth, since no congestion is present on the backward path. However, this time the P2P-TV application suffers for the presence of TCP traffic; note indeed, that the upload bit rate $s(t)$ significantly decreases at about $t = 300$ s, when the first TCP flow starts. In this case, congestion arises in the downstream link. Indeed, since TCP flows use all the available bandwidth, some packet loss and significant queuing delay occur. This makes the peer appear less performing to other peers, that might decide not to select it to download the content. While losses do not significantly affect the video quality (being the application capable of copying with packet loss and additional delay, as seen in the previous section), the large delay prevents the peer from effectively redistributing the chunks it gets.

As a conclusion, the coexistence of P2P-TV and background traffic can be rather critical. On the one hand, peers are still capable of correctly receiving the stream, but they cannot significantly contribute to its re-distribution. As a result upload bandwidth of peers, which is a very precious resource for P2P-TV applications, is wasted. On the other hand, not being elastic, P2P-TV traffic can prevent TCP from properly working, especially in presence of scarce bandwidth, as it is typical of today ADSL setups.

7. Fairness between clients

Finally, in this section, we consider the interaction between two clients of the same application sharing the same access link. The objective is to assess fairness. We therefore perform experiments in which two clients are started at the same time and tuned on the same channel. We then apply a downlink bandwidth limitation profile as the one reported in dashed line of Fig. 14: the bandwidth is progressively reduced to 400 kbps and then progressively increased again. The

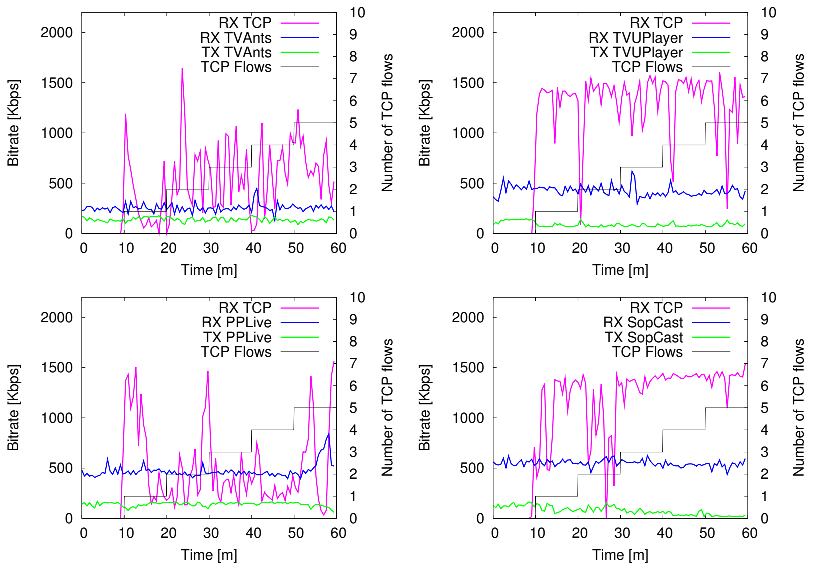

Figure 12: Received and transmitted bit-rate in presence of TCP background traffic. Access link bandwidth parameters: $c(t) = 2$ Mbit/2, $c_u(t) = 200$ kbit/s.

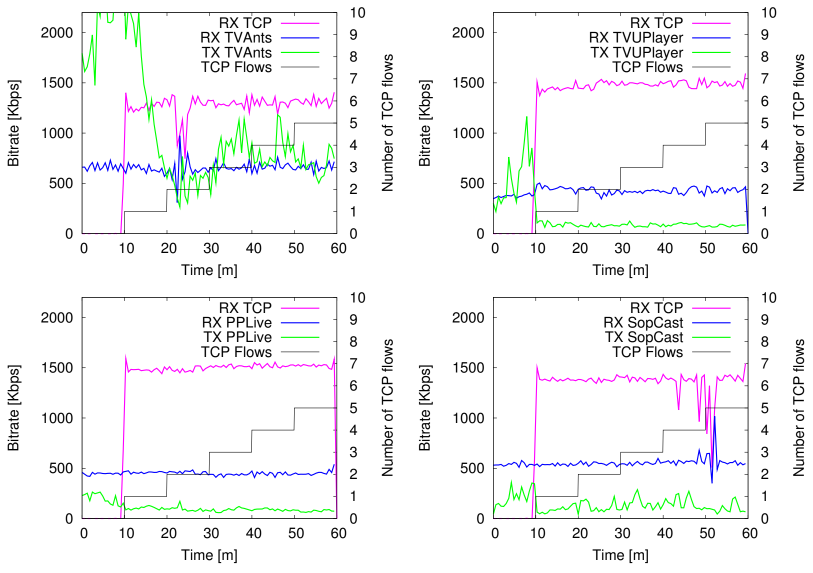

Figure 13: Received and Transmitted bit-rate in presence of TCP background traffic. Access link bandwidth parameters: $c(t) = 2$ Mbit/s, $c_u(t) = 10$ Mbit/s.

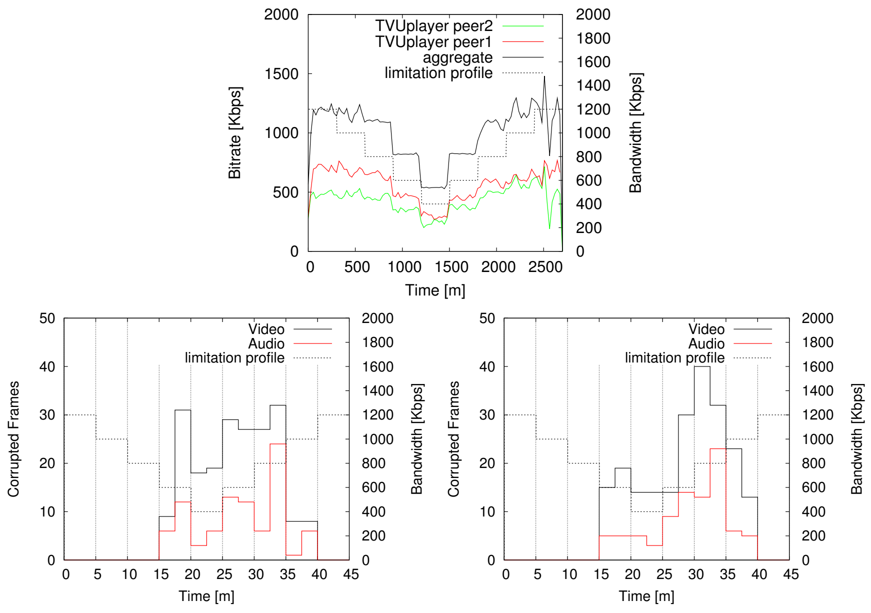

Figure 14: Received bit-rate for two clients concurrently running the same application and sharing the access link (top plot), video and audio impairment for the two clients (bottom plots). TVU-Player and downstream bandwidth limitation.

top plot of the figure refers to TVUPlayer, the red and green lines report the bitrate received by the two clients; the aggregate is reported also, for completeness. A substantial fairness is provided to users, even under severe bandwidth limitation. In these conditions, the video quality degrades (as reported by video and audio impairment on the bottom plots of the same figure) but both the clients undergo the same kind of quality degradation. No real unfairness was observed, even for other kinds of limitation applied through losses or delay, and by starting the clients at different times.

Similar performance and fairness was observed for the other applications too; plots are not reported here for the sake of brevity.

8. Conclusions

In this paper, we propose an experimental methodology to investigate the behavior of P2P-TV applications under adverse network conditions. Since most of the successful P2P-TV applications rely on a closed and proprietary design, it is indeed important to understand if these applications implement algorithms to cope with different and variable network scenarios. In particular, available bandwidth, delay, packet loss probability and presence of background traffic are the most important impairment today applications face in the Internet. We therefore explored how P2P-TV applications react to those parameters, by setting up real test-bed experiments.

We applied this methodology to four P2P-TV applications, namely, PPLive, SopCast, TVAnts and TVUPlayer. By observing the received bit-rate and the number of contacted peers, we have shown that all applications effectively react to impairment caused by: i) lack of bandwidth, ii) packet loss probability, or iii) large delay. Applications indeed successfully select the subset of peers that offer the best performance, disregarding peers on impaired paths. However, in case the bottleneck affects all peers, e.g., it is at the access link, their behavior results rather aggressive, and potentially harmful for both other applications and the network. Interestingly, the control algorithm preferentially operates by selecting the active peers among the neighbors on the overlay, but it does not affect the neighborhood selection, i.e., the overlay topology discovery and setup. The analysis of the upstream traffic has revealed that the peer tries to contribute to the video distribution as far as the network conditions allow it; when the network conditions are bad, the peer acts as a receiver only.

Even if all applications show similar behaviors, some differences arise: TVUPlayer is the fastest and most prompt to react to changing conditions, but sometimes its control algorithm overreacts to dynamic situations; TVAnts, on the contrary, shows a less controlled behavior, which causes the highest overload when resources are scarce, and forces the client to keep downloading from impaired peers.

We have also analyzed scenarios in which P2P-TV applications share the access bandwidth with long lived TCP connections and we have investigated their interaction. In some cases the presence of inelastic P2P-TV traffic may have a negative effect on TCP preventing connections from fully exploiting the available capacity, while in other cases it is the P2P-TV application that suffers the presence of TCP traffic, becoming unable to redistribute the video stream. At last, we have considered clients sharing the same access link and verified that the applications are fair towards these clients even under adverse conditions.

Acknowledgement

This work was funded by the European Commission under the 7th Framework Programme Strep Project "NAPA-WINE" (Network Aware Peer-to-Peer Application under WIse Network).

References

[1] J. Liu, S.G. Rao, B. Li; H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," *Proceedings of the IEEE*, Vol.96, no.1, pp.11-24, Jan. 2008.

[2] A. Murray-Watson, "Internet groups warn BBC over iPlayer plans," *The Independent*, 12 August 2007.

[3] E.Leonardi, M.Mellia, A.Horvart, L.Muscariello, S.Niccolini, D.Rossi, "Building a Cooperative P2P-TV Application over a Wise Network: the Approach of the European FP-7 STREP NAPA-WINE", *IEEE Communications Magazine*, Vol. 46, pp. 20-211, April 2008.

[4] H. Xie, Y. . Yang, A. Krishnamurthy, Y. Liu, A. Silberschatz, "P4P: Provider Portal for Applications", *ACM Sigcomm 2008*, Seattle, WA, August 2008.

[5] D.Bonfiglio, M.Mellia, M.Meo, N.Ritacca, D.Rossi, "Tracking Down Skype Traffic", *IEEE Infocom*, Phoenix, AZ, April 2008.

[6] X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, Vol.9, No.8, pp.1672-1687, Dec. 2007.

[7] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications", *IEEE INFOCOM'08*, Phoenix, AZ, Apr. 2008.

[8] C. Wu, B. Li, S. Zhao, "Multi-channel Live P2P Streaming: Refocusing on Servers" *IEEE INFOCOM'08*, Phoenix, AZ, Apr. 2008.

[9] L. Vu, I. Gupta, J. Liang, K. Nahrstedt, "Measurement of a large-scale overlay for multimedia streaming" *Proc. of the 16th International Symposium on High Performance Distributed Computing,*, Monterey, CA, June 2007.

[10] F. Wang, J. Liu, Y. Xiong, “Stable Peers: Existence, Importance, and Application in Peer-to-Peer Live Video Streaming” *IEEE Infocom’08*, Phoenix, AZ, Apr. 2008.

[11] X. Hei, Y. Liu, K.W. Ross, “Inferring Network-Wide Quality in P2P Live Streaming Systems,” IEEE JSAC, special issue on P2P Streaming, Vol.25, No.9, pp.1640-1654, Dec. 2007.

[12] S. Agarwal, J. P. Singh, A. Mavlankar, P. Baccichet, B. Girod, “Performance and Quality-of-Service Analysis of a Live P2P Video Multicast Session on the Internet,” *IEEE IwQoS*, Enschede, NL, June 2008.

[13] S.Ali, A.Mathur, H.Zhang, “Measurements of Commercial Peer-To-Peer Live Video Streaming,” *In Proc. of Workshop on Recent Advances in Peer-to-Peer Streaming*, Waterloo, ON, Aug. 2006.

[14] T. Silverston, O. Fourmaux, “Measuring P2P IPTV Systems,” *ACM NOSSDAV’07*, Urbana-Champaign, IL, June 2007.

[15] A. Horvath, M. Telek, D. Rossi, P. Veglia, D. Ciullo, M. A. Garcia, E. Leonardi, M. Mellia, “Network Awareness of P2P Live Streaming Applications”, *IEEE Hot-P2P 2009*, Rome, May 2009.

[16] D.Ciullo, M.Mellia, M.Meo, E.Leonardi, “Understanding P2P-TV Systems Through Real Measurements”, *IEEE GLOBECOM 2008*, New Orleans, FL, 30 November 2008.

[17] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, M. Meo, “P2P-TV systems under Adverse Network Conditions: a Measurement Study”, *IEEE Infocom 2009*, Rio de Jainero, Brasil, April 2009.

[18] E. Alessandria, M. Gallo, “P2P-TV Systems Measurements”, Master Thesis, Politecnico di Torino, 2008.